Divide to Conquer: Functional Decomposition to Support Model-Based Engineering of Command and Control of Cyber-Physical Systems

Alexandre Canny, Camille Fayollas, Célia Martinie, David Navarre, Philippe Palanque
ICS-IRIT, University Toulouse 3
Toulouse, France
firstname.lastname@irit.fr

Christine Gris, Yannick Deleris
Airbus
Toulouse, France
firstname.lastname@airbus.com

***Abstract*—Cyber-physical systems (CPS) integrate both physical and computational elements and their engineering requires bridging the continuous analog real world and the discrete digital world. User interfaces of cyber-physical systems belong to the class of command and control systems and their design and engineering usually follow ad-hoc craft processes highly parametrized by the very nature of the physical component(s). This paper proposes a systematic approach for engineering CPS, emphasizing the problems and possible solutions to design and assess their user interfaces (both control and presentation). The paper first proposes a generic architecture for CPS going from the physical element(s) to the user interface. This architecture is then refined, highlighting the behavior and software interfaces of each of its component and showing how information and control from the physical elements have to be processed and transformed to make sense to the operator. The paper presents extracts of the application of the proposed approach to the command and control cockpit application of an aircraft system.**

Keywords—CPS, Interactive Systems, Command and Control

I. INTRODUCTION

While today most of Human-Computer Interaction (HCI) research is targeting at software systems, cyber-physical systems (CPS) deployment has increased in an anarchic way that could be seen a CPS stampede. However, researchers involved in the design, specification, development, validation or maintenance of CPS have been trying to warn about the challenges those systems are bringing. An overview of these challenges can be found in [2] where an interesting (and rather unique) perspective is given towards the need of User Centered Design approaches in that domain.

Taking the Human-Computer Interaction point of view, CPSs have a very specific characteristic: their entire user interface is composed of two relatively independent elements:

- The command and control system designed for allowing operators/users to use the CPS,
- The physical (hardware) part of the CPS that might (or might not) be perceivable or actionable by the user.

While the first aspect is usually the focus of designers and developers, the second aspect is usually ignored even though it might deeply interfere with the actual use of the system. An example of such CPS system is the Philips Hue smart lamp [20] (see *Fig. 1*) which is considered one of the typical IoT devices and the most popular smart light system. While the command and control is located on the remote control and/or on the mobile App, the physical device (the light bulb) is producing both light and heat that are both perceivable by the users. When using the remote control (which is only an input device and has no display on it), the user has to target the connection bridge (not the light bulb) and the only way to know the status of the device is to look at its hardware (physical) part.

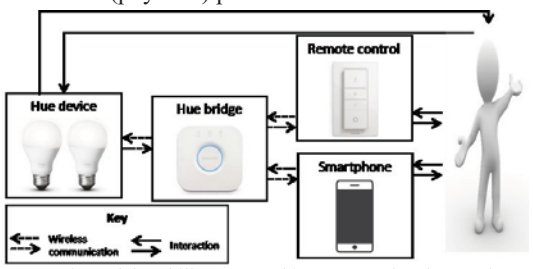

Fig. 1 Overview of the Philips Hue and its command and control systems

The architecture presented in *Fig. 1* shows that, in order to operate appropriately the CPS, users have to be provided with a detailed and complete description of both the hardware and software parts of the CPS but also the connection infrastructure. Indeed, users should point the remote control towards the Hue bridge and not the lamp as most would do (even though new protocols such as Wibree (Bluetooth low energy) don't require direct pointing at the sensor). It is also important to note that the command and control system part (both the smartphone and the remote control) is also a CPS featuring cyber and physical integration. This is the case with the mobile application when users interact with hardware tactile screen for controlling the software part of the App.

Engineering the entire user interface of a CPS thus requires taking into account all CPS elements in an integrated framework to address usability issues.

This paper proposes a framework addressing various aspects of CPS with a specific focus on the human in the loop aspect. This aspect has been explicitly identified in the CPS concept map [12] where this challenged is phrased as "Many cyber-physical systems include humans as an integral components. Humans are very difficult to model, so understanding and validating such systems becomes particularly challenging".

This paper is structured as follows. Section 2 highlights the specificities of user interactions with CPS as well as the need for model-based approaches and presents a list of requirements to model user interactions with CPS. Section 3 presents a generic architecture for engineering the command and control part of CPS. Section 4 presents a set of notation and tools that fulfill the requirements. Section 5 demonstrates the advantages of breaking down CPS complexity using the generic architecture and shows that the modeling techniques are able to cover all the elements of the CPS thanks to extracts from an industrial case study. Section 6 positions the proposed contribution with respect to previous work while section 7 concludes the paper and highlights future directions.

II. Challenges in Engineering the Command and Control of Cyber-Physical Systems

CPS are intended to be used by humans [4] via direct interaction [18] and/or using a monitoring and control user interface [23]. This section presents the main problems that have to be taken into account when engineering user interfaces of the command and control part of a CPS. It also highlights the need for model-based approaches to support the engineering of this type of system. In particular, it presents a set of requirements for the modeling of the command and control part of a CPS.

A. Monitoring, Command and Control of CPS

Our work takes place in the context of large-scale CPS that the user has to monitor and control (e.g. command and control rooms, aircraft cockpit…). However, such kind of problematics is also studied in other application domains such as smart home environments [22]. The user has to perform tasks using the CPS (e.g. switching on the light) but the user is also in charge of monitoring the state of the CPS (e.g. checking whether the light bulb is broken or failing) to ensure that s/he will be able to perform her/his tasks. The two main types of tasks are:

- Mission tasks. They are tasks that have to be accomplished by the user so that s/he can achieved her/his goals.
- Platform monitoring and control tasks. They are the tasks that have to be accomplished to ensure that the current state of the CPS provides support for accomplishing the mission tasks. They include monitoring the physical UI of the CPS as well as monitoring and controlling the CPS using a dedicated user interface.

These two types of tasks are tightly coupled and may interfere with each other. For example, if there is a change in the platform status, e.g. some elements of the CPS are not working properly, the user will have to know whether s/he is able to continue her/his mission, or which part of the mission s/he can continue. S/he will also have to know whether some recovery tasks have to be performed to take back the platform in a state that is acceptable for continuing the mission. Engineering CPS thus requires to be able to describe systematically the user task related to the mission and the user tasks related to monitoring and controlling the platform.

B. The Need for Model-based Approaches

Multiple types of techniques and associated tools for representing the CPS elements are required [1] [8] [13] [22]. This means that a single modeling technique is not able to take into account all the aspects of CPSs. During the design and development process, in order to analyze and to engineer a CPS, dedicated types of techniques and tools are used in accordance with the type of element that is engineered [11]. Different types of representations and artifacts are produced to describe and analyze each element composing the CPS. Moreover, an overall view of the elements composing the CPS is required to describe the relationships between the heterogeneous elements of a CPS, and thus between the representations that have been produced for these elements [1] [22].

A representation of the interactions between the elements of a cyber-physical system is required to ensure consistency and integration of these elements [1] [22]. Service-oriented architecture is suitable to represent these interactions [13] [22].

Current literature reports about model-based approaches for engineering the system and software of CPS but do not explicitly take into account the user interactions with the CPSs [10] [18]. In order to explicitly take into account this aspect, User Centered Design approaches are required to develop cyber-physical systems [1][8]. Note that specific challenges in engineering CPSs arise when explicitly taking into account user interactions with a CPS:

- There is a need for standardization of interaction hardware, a need for toolkits and development processes as well as a need for dedicated prototyping tools [18].
- The possible contexts of use, with their associated relevant or undesirable interactions have to be taken into account at design time [2].

C. Requirements to model user interactions with CPS

We propose a set of requirements for explicitly taking into account the user interactions with a CPS (direct user interaction and/or user interactions with a CPS via a command and control system). Each requirement is given a label (e.g. “Req_ism1” means requirement one for interactive system modeling).

1) Interactive system modelling

Taking into account the interactive software parts of a CPS requires a non-ambiguous description technique of both their interconnection and their inner behavior, and any set of notations able to describe these two aspects can be used while it respects the following set of requirements.

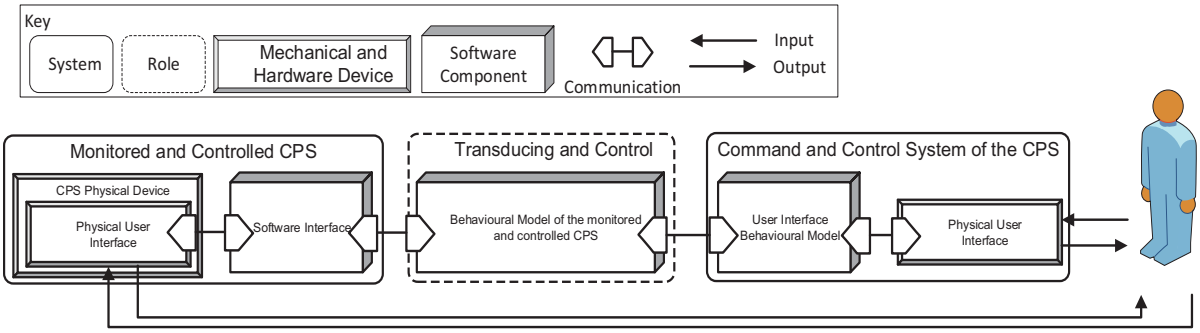

Fig. 2 A generic architecture for supporting the engineering of command and control CPSs.

Requirements for the description of the interconnection between software elements of the CPS:

- Describe the interconnection between the architecture components (*Req_ism1*).
- Describe the interface of each component of the architecture (required methods/services and provided methods/services, incoming and outgoing events) (*Req_ism2*)
- Describe the communication kind between these components (synchronous or asynchronous, unicast or multicast…) (*Req_ism3*)

Requirements for the description of the inner behavior of software elements of the CPS:

- Describe the set of possible actions (*Req_ism4*)
- Describe the set of operator's actions (*Req_ism5*)
- To support the diversity of physical device, the notations must be able to support the description of complex behavior (true concurrency, infinite states, both qualitative and quantitative temporal aspects, etc.) (*Req_ism6*)
- Describe how user actions or inner state changes modify the presentation (*Req_ism7*)
- To be used as part of a user centered design process, it must ease prototyping and user testing (executability, modifiability, etc.) (*Req_ism8*)

2) Operator tasks modeling

Taking into account user activities for the command and control of a CPS as well as the way they may interact with the CPS (directly or indirectly) requires precise and complete description of their activities. Furthermore, in order to be able to analyze potential user errors that may happen at runtime, the following set of requirements has to be match:

- Identify and describe, in a complete and unambiguous way, the user tasks with the CPS, whether they interact in a direct way or with the Command and Control of the CPS (*Req_otm1*)
- Identify and describe the Data (information, objects, knowledge) manipulated by a user for accomplishing her/his tasks (*Req_otm2*)

III. A Generic Architecture to Provide an Overall View on the Elements Composing the CPS

The proposed architecture enables the separation of a complex system into several components. These components are then easier to apprehend. Beside this, these components are strongly consistent and dimly coupled. This allows the separation of concerns (for instance, the components related to the UIs are situated within the UI side) and the locality of modification (a modification within one of the components will not necessarily impact the other components).

Fig. 2 presents a generic architecture, mixing both physical systems and software components, illustrating how the CPS modelling can be broken down into generic components in order to support the model-based engineering of its command and control system. In this Figure, we identified three parts: i) the "Monitored and Controlled CPS", ii) the "Transducing and Control" and iii) the "Command and Control System of the CPS". Each of them is divided in one or more components (either being hardware or software components) with a precise role. The following paragraphs describe each one of these components from left to right and exemplify their usage using the Philips Hue CPS. *Fig. 3* presents the instantiation of our generic architecture to the Philips Hue CPS.

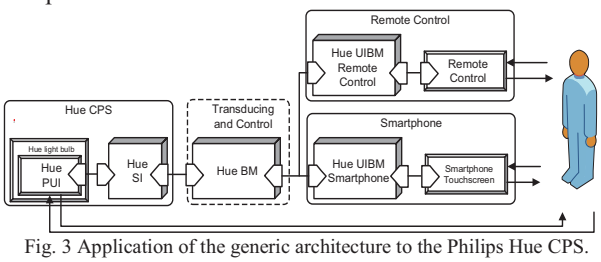

Fig. 3 Application of the generic architecture to the Philips Hue CPS.

A. The Monitored and Controlled CPS

This part corresponds to the CPS device that is monitored and controlled by the user.

Illustration with the Philips Hue CPS: the "Monitored and Controlled CPS" (*Fig. 3*) corresponds to the Hue light bulb.

1) The CPS Physical Device

The CPS physical device is the concrete physical system that has to be operated using a command and control user interface. A detailed description of its inner behavior is not always known and not always mandatory to allow operations. The inner state of such device is usually discovered using sensors while it can be handled using actuators.

Illustration with the Philips Hue CPS: the "Hue light bulb" in *Fig. 3*) corresponds to the physical part of the Hue CPS, i.e. the light bulb device.

2) The Physical User Interface

The physical user interface (PUI) of a CPS represents either the physical phenomena that are not handled by the command and control interactive system but that can be perceived by the operator, or dedicated interactors that can be used for emergency purpose for instance. It is for instance incoming smoke, non-sensed vibrations, explosion or special means to interact with the device such as the physical crank of the landing gear for an aircraft. The operators could have to interact with these physical phenomena and special procedures must be created for such abnormal situation.

Illustration with the Philips Hue CPS. This component (named "Hue PUI" in *Fig. 3*) corresponds to both the light and heat produced by the light bulb that are directly perceivable by the user. Another possible interaction that may be described within this component is the fact that the user may decide to unplug the light bulb.

3) The Software Interface

The software interface (SI) component of the "Monitored and Controlled CPS" part provides a view on how to handle the CPS physical device and how to assess its inner state. This component provides for an external use a set of observable parameters that are the result of the discretization of potentially continuous physical phenomenon (for instance, it could provide an engine speed using sensors). It thus provides a set of observation mechanisms whose kind depends of technologies used and a set of commands that may be performed on the CPS physical device (for instance starting or stopping the device). It can push values of the handled parameters to some listeners using dedicated communication means (unicast or multicast, synchronous or asynchronous) or it can require listener to pull values from it.

Illustration with the Philips Hue CPS. This component (named "Hue SI" in *Fig. 3*) first describes to the set of observation mechanisms allowing to know that the light bulb is ON, OFF, what is its color, what is its intensity. Then, it also describes the fact that the light bulb provides possibilities to handle its state: the possibility to turn it on, to turn it off, to change its color or its intensity…

B. The Transducing and Control

The transducing and control part is composed of a unique component: the behavioral model (BM) of the monitored and controlled CPS. This component is responsible for translating both low-level information from the software interface into higher-level operation centric data (for instance, the availability of an electric service depends of an engine speed that must be above a threshold) and higher level commands from the operators into low-level commands compatible with the device. In other words, this component is responsible for digitizing the behavior of the monitored and controlled CPS (which is, by definition, analog). It is important to note that this component has been defined, within our architecture, as a "role" and not as a system. Indeed, following the type of CPS that will be monitored and controlled, this component may be places within the monitored and controlled CPS system itself, within its command and control system or within a third "intermediate" system.

Illustration with the Philips Hue CPS. This component (named "Hue BM" in *Fig. 3*) represents the discrete behavior of the Hue light bulb. Concretely, the behavioral model both describes the different discrete states of the light bulb (such as "ON", "OFF", "Turning ON", "Turning OFF"…) and the different possibilities for state changes (e.g. how is it possible to switch from the "ON" state to the "OFF" state, passing by the "Turning Off" state).

C. The Command and Control System of the CPS

The command and control system of the CPS corresponds to the system that enables the user to interact with the monitored and controlled CPS. It is important to note that for the same monitored and controlled CPS there may be several command and control systems (see for instance the example of the Philips Hue CPS).

Illustration with the Philips Hue CPS. As the Philips Hue CPS provides two command and control systems (the user can choose to use the remote control or the smartphone), *Fig. 3* presents two different command and control systems for the Hue CPS, respectively named "Remote control" and "Smartphone".

The command and control system is divided in the two following different components.

1) The Behavioral Model of the User Interface (UIBM)

The behavioral model of the user interface (UIBM: User Interface Behavioral Model) deals with:

- the rendering of parameters provided by the behavioral model of the monitored and controlled CPS;
- the activation and control of the user's actions in order to provide commands to the behavioral model of the monitored and controlled CPS.

This component may have some local interaction behavior that does not directly affect the device (for instance, some temporary states during operations before acting on the system). As this behavior may be complex, this component is usually broken down in smaller communicating sub-components.

Illustration with the Philips Hue CPS. For the remote control. This component (named "Hue UIBM - Remote Control" in *Fig. 3*) describes the behavior of the remote control. For instance, it describes the fact that a key of the remote control can have two states ("Pressed" or "Released"). For the smartphone. This component (named "Hue UIBM - Smartphone" in *Fig. 3*) describes the behavior of the application that is running on the smartphone. For instance, this component describes the different modes of the application.

2) The User Interface device

The user interface device (UID) is a set of input and output devices dedicated to interact with the monitored and controlled CPS (for instance, in an aircraft it can be dedicated screens, keyboards, physical buttons or physical rotators). The behavioral model of the User Interface asks

the User Interface device for rendering and actions performed with the input devices are transmitted to and handled by the behavioral model of the user interface.

Illustration with the Philips Hue CPS. *For the remote control.* This component (named "Hue UID - Keys of the remote control" in *Fig. 3*) is the concrete remote control device: it is thus composed of the different keys of the remote control. *For the smartphone.* This component (named "Hue UID - Smartphone touch screen" in *Fig. 3*) is the concrete device with which the user can interact: it is thus the smartphone touchscreen.

IV. A MULTI-MODELS BASED APPROACH TO DESCRIBE THE DIFFERENT TYPES OF ELEMENTS IN THE CPS

We here propose a set of notations that fulfill the requirements to model user interactions with CPS (which have been identified in section II.C). Engineering large-scale systems requires being able to describe a large amount of information, computer-aided software tools are thus required.

A. Notations and tools

1) Interconnection between software elements (Req_ism1, Req_ism2, Req_ism3)

The description of the interconnection between components is performed using the CORBA Component Model [17] that allows the description of the kind of communication (event or method call based) and the observable parameters through the definition of attributes.

2) Inner behavior of software elements (Req_ism4 to Req_ism8))

ICO (Interactive Cooperative Object) is a formal description technique based on Petri nets and dedicated to the specification of interactive systems [16]. It uses concepts borrowed from the object-oriented approach (dynamic instantiation, classification, encapsulation, inheritance, client/server relationship) to describe the structural or static aspects of systems, and uses high-level Petri nets [6] to describe their dynamic or behavioral aspects. ICO is dedicated to the modeling and the implementation of event-driven software components, using several communicating objects to model the system, where both behaviors of objects and communication protocol between objects that are described by Petri nets.

PetShop (Petri Net workshop) is a tool for creating, editing, simulating and analyzing system models using the ICO (Interactive Cooperative Objects) notation [16]. Petshop provides means to analyze ICO models through the analysis of the underlying Petri net model. The PetShop tool provides the means to analyze ICO models by the underlying Petri net model [7] using static analysis techniques as supported by the Petri net theory [19]. The ICO approach is based on high level Petri nets. As a result, the analysis approach builds on and extends these static analysis techniques. It is thus possible to check well-formedness properties of the ICO model, such as absence of deadlocks, as well as user interface properties, either internal properties (e.g., reinitiability) or external properties (e.g., availability of widgets). Note that it is not possible to express these user interface properties explicitly – the analyst needs to express these properties as structural and behavioral Petri net properties that can be then analyzed automatically in PetShop.

3) Precise and complete description of user activities (Req_otm1, Req_otm2)

HAMSTERS (Human centered Assessment and Modeling to Support Task Engineering for Resilient Systems is a task modeling notation. It provides support to describe and structure users' goals and sub-goals into hierarchical task trees. Qualitative temporal relationships among tasks are described by operators. Various notational elements support modeling of specialized task types, explicit representations of data and knowledge.

HAMSTERS is also the name of tool for editing and simulating HAMSTERS task models [14]. It provides support for describing and structuring a large amount of user tasks. These structuring mechanisms enable the breakdown of a task model in several ones that can be reused in the same or different task models. The HAMSTERS task modeling tool provides support for creating, editing, and simulating the execution of task models.

B. Process

There is no unique process to follow in order to model the various elements of the proposed architecture. Depending on the purpose of the engineering of the command and control of the CPS, some architecture components might be considered first:

- When designing the entire CPS (for instance when creating a new one), typically the flow of modeling would be from the physical system to the user interface part (left to right on the architecture in *Fig. 3*). This flow of modeling was applied in the case of the analysis of the command and control of the APU;
- When designing a new User Interface (UI) of the command and control of an existing CPS, the flow of modeling would focus on the UI models leaving untouched the other ones. In this case the list of services provided by the other elements of the CPS are considered as input for the design of the UI.

V. EXTRACTS FROM THE APPLICATION OF THE APPROACH TO AN AIRCRAFT SYSTEM

We here present extracts of models that have been produced during the analysis of an existing command and control UI for aircraft systems. In this example, we focus on one system: the Auxiliary Power Unit (APU).

The APU is a turbine that enables an aircraft to be autonomous regarding electrical power and bleed air. The pilots can interact with it through a user interface made up of two distinct parts: the overhead control panel and the control and display system. Depending on the flight phase, if enabled, the APU can provide: bleed air for engine start and air conditioning and electrical power, bleed air to assist

engine start, bleed air as a backup for pressurization and air conditioning and backup for electrical power.
As presented in *Fig. 4*, the APU needs electrical and fuel resources and delivers electricity and bleed to the aircraft.

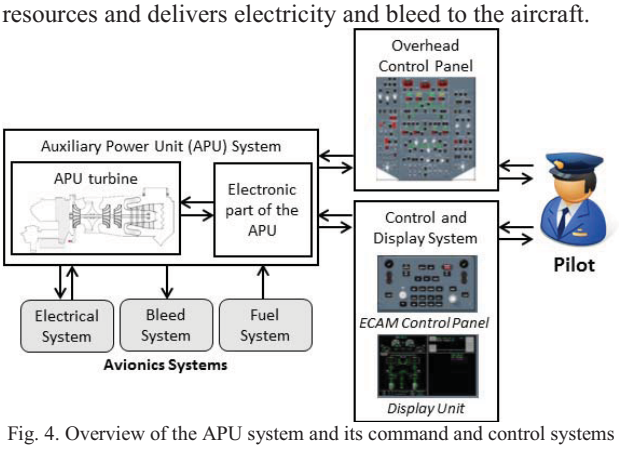

Fig. 4. Overview of the APU system and its command and control systems

The pilots can interact with the APU through two different user interfaces: the overhead control panel and the display unit (see *Fig. 4*).
Fig. 7 presents the application of the generic architecture to the command and control of the APU system. As mentioned in the introduction (using Philip Smart Hue lamp), the physical part of the cyber physical system is also perceivable by the operator. In the case of the APU CPS, output information such as vibrations, smoke and noise can be perceived by the pilot (as illustrated by the arrows between the user and the Physical User Interface component of the APU in *Fig. 7*).

A. *Extract from the representation of the interconnection between software elements*

Fig. 5 presents the abstract component based representation of the "APU SI " element.

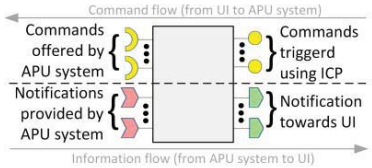

Fig. 5. Abstract Corba representation of the APU HW Interface

The upper part of the component corresponds to the command flow from the user interface to the APU system, while the lower parts correspond to the information about the APU system that has to be presented on the user interface. *Fig. 6* presents an excerpt of the concrete representation of the element "APU SI ".

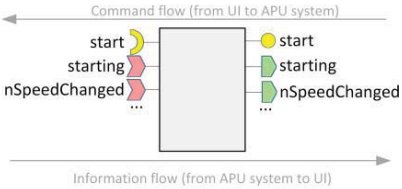

Fig. 6. Concrete representation of a subset of the APU HW Interface

When the operator presses the "Start APU" button, the event flows through each element of the architecture and is received by the "APU SI" component (start facet on top right-hand side in *Fig. 6*). In turn, this component triggers the starting of the APU physical device start receptacle (on top left-hand side in *Fig. 6*).

B. *Extract from the representation of the inner behavior of the software elements*

Fig. 8 presents an extract of the ICO model of the behavioral model of the APU ("APU BM" in *Fig. 7*). The first ICO model of the "APU BM" ICO model (*Fig. 8* a)) corresponds to the switching from APU stopped to APU started. The two places in *Fig. 8* a) (round shapes) correspond to two different states of the APU device: when a token is present within the "FlapOpened" place (as it is the case in Figure *Fig. 8* a), the APU device is ON. When the APU device starts, the "APU SI" sends the "starting" event. This event (if the "APU BM" has a token in the "FlapOpened" place) triggers the firing of the "switchToStarting" transition (rectangle shape). The firing of this transition first sends the "apuStarting" higher-level event (that can be used by the "APU UIBMs") and second consumes the token in the "FlapOpened" place and put a new token in the "Starting_ElecRequired" place.
The second ICO model of the "APU BM" ICO model (*Fig. 8* b)) corresponds to the switching from APU starting to APU started. In *Fig. 8* b), a token is present in the "Starting_FuelRequired" place. When the APU device rotation speed increases, the "APU SI" sends "nSpeedChanged" events. These events are containing a parameter named "nSpeed" corresponding to the value of the rotation speed. When this parameter is superior or equal to 93, the "switchToAvail" transition is fired (as stated by its event condition).

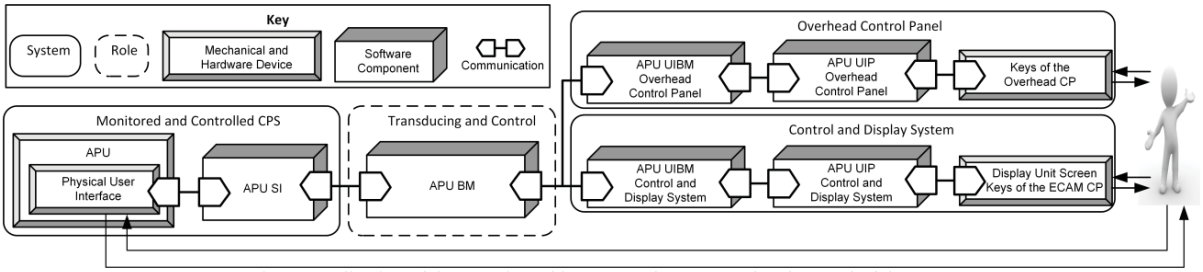

Fig. 7. Application of the generic architecture to the command and control of the APU

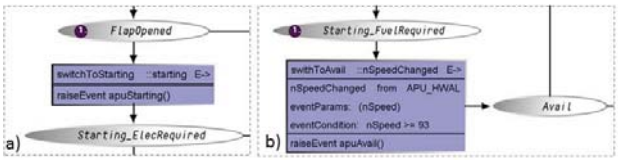

Fig. 8. Excerpt of the "APU BM" ICO model

Fig. 9 presents an extract of the ICO model of the behavioral model of the User Interface of the overhead control panel. This extract is the model of the handling of user inputs using the "START" pushbutton of the overhead control panel. When the user presses this pushbutton, the overhead control panel device sends the "StartPbPressed" event, which triggers the firing of the "pressStartPb" transition. When the user releases this pushbutton, the overhead control panel device sends the "StarPbReleased" event that triggers the firing of the "apuStart" transition. The firing of this transition implies that the "start" method is called on the "APU BM" ICO model ("apu.start()" command within the "apuStart" transition). When the "APU BM" model receives this method call, it transfers it to the "APU SI" that will ask the APU system to start.

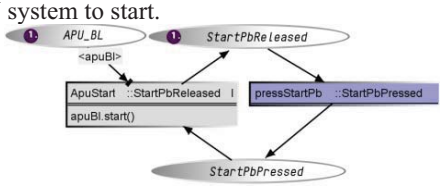

Fig. 9. Excerpt of the "APU UIBM overhead control panel" ICO model: handling of user inputs with the "START" PushButton

C. Extract from the representation of the user activities

Fig. 10 presents an extract of the task model describing a set of user actions that are the first step to start the APU (named "Perform Master sw" because the Master sw is a button that needs to be pushed to enable the starting of the APU).

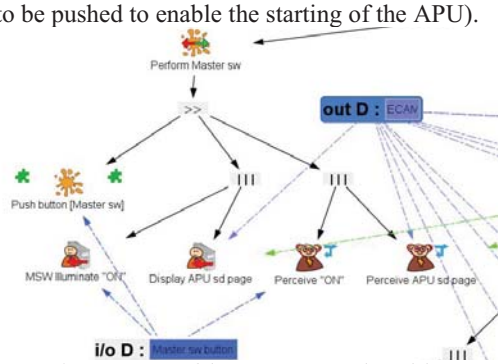

Fig. 10. Extract of the "Start APU" task model

The user first has to push the "Master sw" button, then the button illuminates (interactive output task labeled "MSW illuminate "ON"). Simultaneously, the APU sd page is displayed in the command and control interface (interactive output task labeled "Display APU sd page"). Ultimately, the user perceives that the button is ON (user perceptive task labeled "Perceive "ON"") and perceives the sd page displayed (user perceptive task labeled "Perceive APU sd page").

VI. Related work

A. Prototyping and Validation Approaches for Engineering Command and Control of CPS

Mueller et al. [15] have coined a toolset for virtual prototyping of CPS that provides support for modeling the whole set of components composing a CPS: from physical environment to the monitoring and control user interface, including the modeling of mechanical hardware, electronics and software. It aims at reaching a time performance that is close to the real life usage of the analyzed CPS, in order to provide support for verification and validation. Schirner et al. [21] propose a prototyping environment for exploring interactions between users and assistive robots, in order to provide support for augmenting user interactions with the physical world. These kind of approaches are not model-based and the produced artefacts cannot be directly reused in a development process. Tan et al. [24] present the principle of a prototype architecture for CPSs which covers some aspects not covered in this paper (for instance the elements dealing with security). However, [24] misses the point that the human might be in control of the CPS and that the CPS "raison d'être" might be to support operators in the performance of their tasks.

B. Model-based Engineering the Specification and Development of Command and Control of CPS

Seiger et al. [22] propose a business process modeling approach to design and develop workflows in smart environments, based on UML and Petri nets modeling notations. Bhave et al. [1] focus on representing a CPS and on the importance of having a central representation for connecting the different required types of models. Franke et al. [5] focus on the middleware components of smart home environments. They propose a semantic and model-based approach for specifying and executing the middleware components of CPS, where devices can be added or removed by a user. However, these approach does not provide support for engineering user interactions with the CPS. Jensen et al. [9] propose a ten steps method for designing a CPS and to ensure that requirements are met. This approach details the modeling of the physical components but does not provide insights neither on software specification and modeling, nor about how the different types of models may be integrated. Yue et al. [25] show that event-based approaches provide support for ensuring consistency between different levels of events generated by the different components of a CPS. They propose an event based modeling technique for specifying the behavior of a CPS but do not provide information about how these models can be integrated with other types of models such as physical, electronic or user interface ones.

VII. Conclusion and Future Work

The proposed generic architecture and model-based approach gives a holistic view of the CPS and enables the integration of the UI models of the CPS (e.g. the "APU UIBMs" in the illustrative example) with the system models of the CPS (e.g. the "APU SI" and the "APU BM" in the illustrative example) and thus provides support for:

- Specifying the components of the APU CPS;
- Ensuring consistency between all of the components of the CPS, and then between the UIs of the CPS and all of the other components of the CPS;
- Producing high-fidelity prototypes of the CPS and of its UIs;
- Validating the behavior of each of the components of the CPS, as well as its behavior as a whole.

In addition, the proposed architecture and its associated models provide support for ensuring consistency between the context of use (e.g. the activities that have to be performed by the pilots and in which contexts) and the possible interactions.

While this article focuses on providing support for analyzing usability of the CPS, other properties require to be taken into account. For example, the proposed approach supports the assessment of the reliability of the CPS if using a formal notation for the description of the CPS behavior. We argue that there is a clear need to follow a holistic approach building on previous work done in domains such as dependability [1] and security [3], and to integrate them.

References

[1] S. Abdelwahed, N. Kandasamy, and A. Gokhale. 2007. High Confidence Software for Cyberphysical Systems. In Proceedings of the 2007 Workshop on Automating Service Quality: Held at the International Conference on Automated Software Engineering (ASE) (WRASQ '07). ACM, New York, NY, USA, 1–3.

[2] M. Broy and A. Schmidt. 2014. Challenges in Engineering Cyber-Physical Systems. Computer 47, 2 (Feb 2014), 70–72.

[3] Á. A. Cárdenas. 2014. From CRCs to Resilient Control Systems: Differentiating Between Reliability and Security for the Protection of Cyber-physical Systems. In Proceedings of the 3rd International Conference on High Confidence Networked Systems (HiCoNS '14). ACM, New York, NY, USA, 125–126.

[4] Cyber-Physical Systems Virtual Organization Home page. http://cps-vo.org/. Last accessed on Feb 20th 2019.

[5] M. Franke, C. Seidl, and T. Schlegel. 2013. A seamless integration, semantic middleware for cyber-physical systems. In 2013 10th IEEE INTERNATIONAL CONFERENCE ON NETWORKING, SENSING AND CONTROL (ICNSC). 627–632.

[6] H. J. Genrich. 1991. Predicate / Transition Nets. Springer Berlin Heidelberg, Berlin, Heidelberg, 3–43.

[7] Arnaud Hamon Philippe palanque Célia Martiinie Eric Barboni José-Luis Silva, Camille Fayollas. 2014. Analysis of WIMP and Post WIMP Interactive Systems based on Formal Specification. Electronic Communications of the EASST 69, Article 55 (2014), 29 pages.

[8] R. Harrison, D. Vera, and B. Ahmad. 2016. Engineering Methods and Tools for Cyber-Physical Automation Systems. Proc. IEEE 104, 5 (May 2016), 973–985.

[9] J. C. Jensen, D. H. Chang, and E. A. Lee. 2011. A model-based design methodology for cyber-physical systems. In 2011 7th International Wireless Communications and Mobile Computing Conference. 1666–1671.

[10] S. K. Khaitan and J. D. McCalley. 2015. Design Techniques and Applications of Cyberphysical Systems: A Survey. IEEE Systems Journal 9, 2 (June 2015), 350–365.

[11] Edward A. Lee. 2016. Fundamental Limits of Cyber-Physical Systems Modeling. ACM Trans. Cyber-Phys. Syst. 1, 1, Article 3 (Nov. 2016), 26 pages.

[12] Edward A. Lee Martin S. Shyam Sunder Philip Asare, David Broman. 2017. Cyber-Physical Systems - a Concept Map. In Web only. Last accessed on Feb 20th 2019, https://cyberphysicalsystems.org/

[13] Chang Hong Lin, Marilyn Wolf, Xenefon Koutsoukos, Sandeep Neema, and Janos Sztipanovits. 2010. System and Software Architectures of Distributed Smart Cameras. ACM Trans. Embed. Comput. Syst. 9, 4, Article 38 (April 2010), 30 pages.

[14] C. Martinie, P. Palanque, and M. Winckler, "Structuring and Composition Mechanisms to Address Scalability Issues in Task Models," in Human-Computer Interaction – INTERACT 2011, 2011, pp. 589–609.

[15] W. Mueller, M. Becker, A. Elfeky and A. DiPasquale, "Virtual prototyping of Cyber-Physical Systems," 17th Asia and South Pacific Design Automation Conference, Sydney, NSW, 2012, pp. 219-226.

[16] David Navarre, Philippe Palanque, Jean-Francois Ladry, and Eric Barboni. 2009. ICOs: A Model-based User Interface Description Technique Dedicated to Interactive Systems Addressing Usability, Reliability and Scalability. ACM Trans. Comput.-Hum. Interact. 16, 4, Article 18 (Nov. 2009), 56 pages.

[17] Object Management Group CORBA Components OMG Document formal/2002-06-65. (June 2002).

[18] Volker Paelke and Carsten Rocker. 2015. User Interfaces for Cyber-Physical Systems: Challenges and Possible Approaches. Springer International Publishing, Cham, 75–85.

[19] J. Peterson, Petri Net Theory and the Modeling of Systems. Prentice Hall PTR, 1981.

[20] Philips, Hue lamp. http://www2.meethue.com/en-us/products/. Last accessed on Feb 20th, 2019.

[21] Gunar Schirner, Deniz Erdogmus, Kaushik R. Chowdhury, and Taskin Padir. 2013. The Future of Human-in-the-Loop Cyber-Physical Systems. IEEE Computer 46, 1 (2013), 36–45.

[22] Ronny Seiger, Christine Keller, Florian Niebling, and Thomas Schlegel. 2015. Modelling complex and flexible processes for smart cyber-physical environments. Journal of Computational Science 10 (2015), 137 – 148.

[23] Ronny Seiger, Diana Lemme, Susann Struwe, and Thomas Schlegel. 2016. An Interactive Mobile Control Center for Cyber-physical Systems. In Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing: Adjunct (UbiComp '16). ACM, New York, NY, USA, 193–196.

[24] Ying Tan, Steve Goddard, and Lance C. Pérez. 2008. A Prototype Architecture for Cyber-physical Systems. SIGBED Rev. 5, 1, Article 26 (Jan. 2008), 2 pages.

[25] K. Yue, L.Wang, S. Ren, X. Mao, and X. Li. 2010. An adaptive discrete event model for cyber-physical system. In Proc. of Analytic Virtual Integration of Cyber-Phys. Syst. Workshop. 9–15